# APEX Trading API Development Guide

V1.4

## Review History:

| Version | Date | Changes |
|---|---|---|
| 1.0 | 29 June 2018 | First Draft |
| 1.1 | 20 Dec 2018 | • Added “Order Management” subsection<br>• Added “Local Files” subsection |
| 1.2 | 4 Sep 2019 | Added “Order Types” section |
| 1.3 | 6 Nov 2019 | Added “Heartbeat” section |
| 1.4 | 20 Jan 2020 | Added “Spread” section |

## Table of Contents

## 1 Objective

The objective of this guide is to provide best programming for systems that interact with APEX trading platform.

## 2 Development Guides

### 2.1 Login

- **Same login for Trading and MD**

  The same login ID can be used to login both Trading Gateway and Market Data Gateway concurrently. However, if the same login is used to login to the same gateway again, the existing login will be forced to disconnect.

- **Use Name Server for login**

  Although APEX API supports login through both Name Server and Gateway Server, only login through Name Server is advised. This is because during failover, Name Server will return the new Gateway Server IP for TraderAPI to login so that failover will be transparent for users.

  There are four Name Servers in production: two in primary and two in DR (Disaster Recovery). User should always try the two primary Name Servers first followed by the two in DR. Below is a sample of such implementation (please check with APEX on latest Name Server IPs):

  - ❖ Trading:

```
char *ns[] = {
  "tcp://10.32.100.31:4901",
  "tcp://10.32.100.32:4901",
  "tcp://10.32.100.33:4901",
  "tcp://10.32.100.34:4901"
}
for (int i = 0; i < 4; i++) {
  pTraderApi->RegisterNameServer(ns[i]);
}
```

  - ❖ Market Data:

```
char *ns[] = {
  "tcp://10.32.100.31:4903",
  "tcp://10.32.100.32:4903",
  "tcp://10.32.100.33:4903",
  "tcp://10.32.100.34:4903"
}
for (int i = 0; i < 4; i++) {
  pMdApi->RegisterNameServer(ns[i]);
}
```

- **Retry upon Login Error**

  APEX will advise on login/logout timings and users are advised to login before the suggested login time and logout after the suggested logout time. However, if the user logins too early before Trading Engine is ready, following error may be returned:

  - ErrorID = 45
  - ErrorMsg = "INVALID DATA GROUP DATASYNC STATUS IN INITIALIZATION"

  User is advised to retry login every 1 minute until Trading System is ready with successful login return code, to prevent failed login.

- **User ID Type**

  There are two types of User ID:

  - Single Trade: only able to receive order/trades initiated by current login user.
  - Manage Trade: able to receive order/trades initiated by all login users under the current Member ID and its sponsored Member ID(s).

  In production, only "Single Trade" will be issued unless "Manage Trade" is specially requested and approved.

- **Subscription Type**

  Public Topic should always be subscribed to receive instrument status updates. Single Trade User should also subscribe User Topic to receive orders and trades initiated by the user himself. On the other hand, Manage Trade User should also subscribe Private Topic to receive orders and trades initiated by all users under the same Member.

- **Local Files**

  During runtime, the API generates local files with ".con" file extension containing connection information. The path of the local files can be specified by calling *CreateFtdcTraderApi* method. Different users must specify different paths, otherwise they may not be able to receive some data from the Trading System.

## 2.2 Logout

Although user can call *ReqUserLogout* method to logout, API will automatically re-connect, as a mechanism to maintain connectivity. So in order to logout and disconnect, user needs to explicitly terminate the connection inside *OnRspUserLogout* method.

## 2.3 Heartbeat

The API and APEX trading system exchange heartbeat messages to monitor the network connection status. If the API does not receive a heartbeat message after a certain delay, it will invoke *OnHeartBeatWarning* callback. This may occur due to network disruption or busy API thread (caused by long processing time in callback methods). Hence, please do not include long time processing in the API callback functions, to avoid the heartbeat warning and unexpected disconnection. The default heartbeat timeout is 10 seconds. User can change the heartbeat timeout by calling *SetHeartbeatTimeout* method.

## 2.4 Disconnection handling

Upon disconnection and re-login, user needs to take care of following scenarios correctly to avoid missing or redundant data:

- If *APEX_TERT_RESTART* mode is used upon re-login, since all orders/trades since beginning of the day will be resent, user need to be able to identify and ignore duplicate orders/trades/MD (by unique identifiers like OrderSysID, OrderLocalID and TradeID).
- If *APEX_TERT_RESUME* mode is used upon relogin, and there is data received before disconnection but no processed (due to client application core dump or malfunction), there could be missing orders/trades. This mode should not be used if it is not guaranteed that all data will be processed.
- *APEX_TERT_QUICK* is not advised to use since it does not tell whole order history.

## 2.5 Order handling

- **Retrieve *MaxOrderLocalID* upon login**

User should retrieve *MaxOrderLocalID* after successful login in *OnRspUserLogin()*, which should be used as the base of new *OrderLocalID* and *ActionLocalID*.

- **Monotonically Increasing *OrderLocalID* and *ActionLocalID***

To increase the *OrderLocalID* every time in order messages, a recommended way is to combine member identifier, trading day, timestamp, and an increasing order number as *OrderLocalID*. *ActionLocalID* should be managed the same way.

  - *OrderLocalID* and *ActionLocalID* are of String type and share the same series, which means next *OrderLocalID* or *ActionLocalID* must be greater than *Max[last OrderLocalID, last ActionLocalID]*.

- **Order Management**

Night session orders are carried over to the morning session and afternoon session of the same trading day. Within a trading day, each order can be uniquely identified by its OrderSysID. OrderSysID is reset every trading day, which means that orders from different trading days may have the same OrderSysID.

- **Order Deletion upon Market Close**

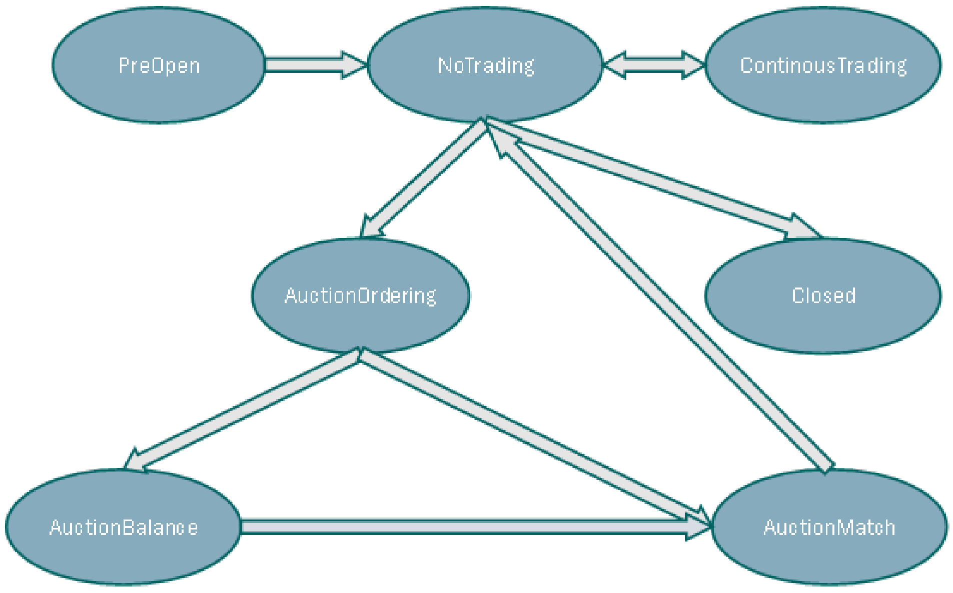
During trading pause, instrument status switches to NoTrading (`APEX_FTDC_IS_NoTrading '1'`). Orders should be kept in order book so they can resume when next session starts. At the end of trading day (normally at the end of afternoon session), instrument status switches to Closed

(APEX_FTDC_IS_Closed '6') and all active orders should be deleted on client. Exchange will run end of day cycle and delete all client's orders. Clients should not send API calls to delete their orders since these calls will be rejected due to Market close.

- **Allowed Actions for each Instrument Status**:

| **Instrument Status Enum** | **Order Allowed?** | **Matching Allowed?** | **Notes** |
|---|---|---|---|
| APEX_FTDC_IS_BeforeTrading | No | No | Ready for API connection |
| APEX_FTDC_IS_NoTrading | No | No | Market Pause during trading sessions within a complete trading day. |
| APEX_FTDC_IS_Continous | Yes | Yes | Normal Trading |
| APEX_FTDC_IS_AuctionOrdering | Yes | No | Pre-opening |
| APEX_FTDC_IS_AuctionBalance | No | No | (Not in use) |
| APEX_FTDC_IS_AuctionMatch | No | Yes | Auction Matching |
| APEX_FTDC_IS_Closed | No | No | Market Close at the end of whole trading day. |

- **Flow diagram of Instrument Status**:



## 2.6 Order Types

| Price Condition | Volume Condition | Time Condition | Supported | Remarks |
|---|---|---|---|---|
| **Limit Price** | AllVolume | GFD | No | Unsupported order type |
| | | IOC | Yes | FOK |
| | AnyVolume | GFD | Yes | Typical "GFD" |
| | | IOC | Yes | FAK |
| | MinVolume | GFD | No | Volume constraint can only apply to IOC order. |
| | | IOC | No | FAK with minimum filled volume |
| **Any Price** | AllVolume | GFD | No | Unsupported order type |
| | | IOC | No | Unsupported order type |
| | AnyVolume | GFD | No | "Market to Limit" The remaining quantity will convert to the last price and rest in order book. |
| | | IOC | Yes | Typical "IOC" |
| | MinVolume | GFD | No | Unsupported order type |
| | | IOC | No | Unsupported order type |
| **Best Price** | AllVolume | GFD | No | Unsupported order type |
| | | IOC | No | Unsupported order type |
| | AnyVolume | GFD | No | "Market to Limit" The remaining quantity will convert to the counter price and rest in order book. |
| | | IOC | No | The remaining quantity will be cancelled. |
| | MinVolume | GFD | No | Unsupported order type |
| | | IOC | No | Unsupported order type |
| **Five Level Price** | AllVolume | GFD | No | Unsupported order type |
| | | IOC | No | Unsupported order type |
| | AnyVolume | GFD | No | "Market to Limit" The remaining quantity will convert to the last price and rest in order book. |
| | | IOC | No | The remaining quantity will be cancelled. |
| | MinVolume | GFD | No | Unsupported order type |
| | | IOC | No | Unsupported order type |

Note: only GFD (Good For Day) and IOC (Immediate or Cancel) are available, other TimeConditions are all not effective, including:

1) GFS: Good For Session

2) GFA: Good For Auction
3) GTD: Good Till Date
4) GTC: Good Till Cancel

Only normal GFD orders are allowed in AuctionOrdering. Order entering is not allowed in AuctionMatching.

## 2.7 Market Data handling

- **Handling no data**

  DBL_MAX is sent if a data field of floating type is empty, it should be displayed as 0 (for volume) or empty in front end. This value should be checked before further processing:

```
double format_price(const double price)
{
        return is_type_max<double>(price) ? 0.0f : price;
}
```

- **DSP**

  DSP will be sent out after market close.

## 2.8 Spread Trading

The Spread Trading is now supported by APEX Market Data and Trading API, please filter out the spread instruments and market data updates if spread trading is not supported by your platform.

For more information of the Spread Trading, please see the details below.

- **Supported Spread Types**

  The current supported Spread Types are Calendar Spread and Inter-Commodity Spread.

  A Calendar Spread is a futures or options spread established by simultaneously entering a long and short position on the same underlying asset but with different contract months.

  An Inter-Commodity Spread is a futures or options spread established by simultaneously entering a long and short position on the different underlying asset but with the same contract month.

- **Query Spread Instruments**

  The Spread instruments can be queried/downloaded from either market data or trading APIs using **ReqQryInstrument** and **ReqQryCombinationLeg**.

  The **ReqQryInstrument** method is used to query all the current listed instruments. The Spread instruments' ProductClass is

`APEX_FTDC_PC_Combination('3')`, while the Future instruments' ProductClass `is APEX_FTDC_PC_Futures('1')`.

The **ReqQryCombinationLeg** method is used to query all the leg information of the Spread instruments. A Spread instrument has three legs – one for the Spread itself, and the other two are for its two legs.

For example, the Future Calendar Spread SP_PF2003&PF2004 has two future legs PF2003 and PF2004, see the below table for more details.

| **Spread Instrument ID** | **Leg ID** | **Leg Instrument ID** | **Leg Direction** | **Leg Multiple** | **ImplyLevel** |
|---|---|---|---|---|---|
| SP_PF2003&PF2004 | 0 | SP_PF2003&PF2004 | 1 | 1 | 1 |
| SP_PF2003&PF2004 | 1 | PF2003 | 0 (Buy) | 1 (Ratio) | 1 |
| SP_PF2003&PF2004 | 2 | PF2004 | 1 (Sell) | 1 (Ratio) | 1 |

Spread Instrument ID: the spread instrument ID.

Leg ID: 0 indicates the Spread itself, 1 and 2 indicate the first and second legs.

Leg Instrument ID: the leg instrument ID.

Leg Direction: the leg side. 0 indicates "buy", 1 indicates "sell".

Leg Multiple: the leg ratio. The value is always 1 for Calendar Spread.

Imply Level: Not in use for now, please ignore.

- **Determine Spread Type**

  The current supported Spread Types are Calendar Spread (e.g. SP_PF2003&PF2004) and Inter-Commodity Spread (e.g. SPC_CPF2003&PF2003). There are two ways to determine the Spread Type.

  A quick way is to determine the Spread Type from the instrument ID directly. For example, a Calendar Spread instrument ID is prefixed with "SP_", whereas an Inter-Commodity Spread instrument ID is prefixed with "SPC_". Currently this is acceptable as APEX only supports the two Spread Types.

  Another better way is to determine the Spread Types from its legs. A Spread instrument is a Calendar Spread if its two future legs are from the same underlying commodity, and it is an Inter-Commodity Spread if its two futures legs are from different underlying commodities but with the same contract month.

  It is recommended to use the second way because the Spread instrument ID may not always be generated using the same pattern in the future.

- **Auction Session**

  The Spread orders are not supported in the Auction session. If there are multiple Auction sessions and there are some Spread orders left in the second or later Auction sessions, the matching engine will cancel these Spread orders automatically at the end of the Auction sessions.

- **Spread Orders**

  The available spread order types are **Market**, **Limit**, **Limit-FAK** and **Limit-FOK**, which are the same as those on future instruments.

  The API method **ReqOrderInsert** is used to send Spread orders by filling in the spread instrument ID in the request message. The API method **OnRspOrderInsert** is called as a response after the new Spread orders are sent out to the APEX Exchange. The API methods **OnRtnOrder** and **OnRtnTrade** will be invoked once Spread orders are accepted and executed later.

  The API method **ReqOrderAction** is used to cancel Spread orders. The API method **OnRspOrderAction** is called as a response after the cancel request is sent out to the APEX Exchange. The API method **OnRtnOrder** will be invoked once Spread orders are successfully cancelled later.

  Please note that there are three **OnRtnTrade** API calls once a trade is executed on a Spread order, one is for the spread trade with synthetic trade price, the other two are for the leg trades.

- **Implied Prices**

  Both Implied-In and Implied-Out prices are supported, but only the first level implied prices are shown in the market prices.

  An Implied-In price is a spread price generated from two outright prices from its legs, while an Implied-Out price is an outright price generated from the prices on the spread instrument and the other leg instrument.

  The APEX matching engine disseminates maximum five levels of bid and ask prices. If implied prices are among them, the matching engine will populate the best bid and ask implied prices into the BestBidImpliedPrice, BestBidImpliedVolume, BestAskImpliedPrice and BestAskImpliedVolume fields of the CApexFtdcDepthMarketDataField message.

- **Price Limits**

  The Spread price limits are determined by the price limits of its legs. For example, the price limits of a Calendar Spread instrument are calculated in the following way:

  - Upper Limit Price = Leg 1 upper limit price – Leg 2 lower limit price
  - Lower Limit Price = Leg 1 lower limit price – Leg 2 upper limit price

  Please note that the spread price can be zero or negative, depending on the difference of its leg instrument prices.

- **Instrument Status**

  The Spread instrument status is determined by its leg instrument statues. Spread instruments can have Auction sessions, but spread orders are not allowed in the Auction sessions.

- **Daily Settlement Prices (DSP)**

The DSP are also available on the Spread instruments, which are determined by the difference of its legs' daily settlement prices.